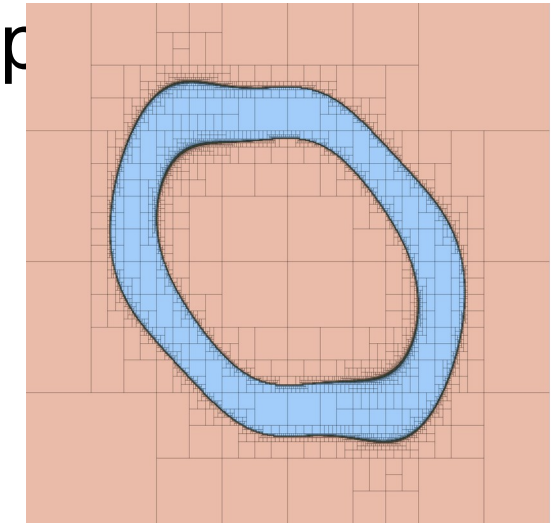
Accelerating the SIVIA algorithm with GPUs

Ander Gray
UTC Compiègne

Why?

- Interval methods often have exponential comp
- Limits the size of solvable problems

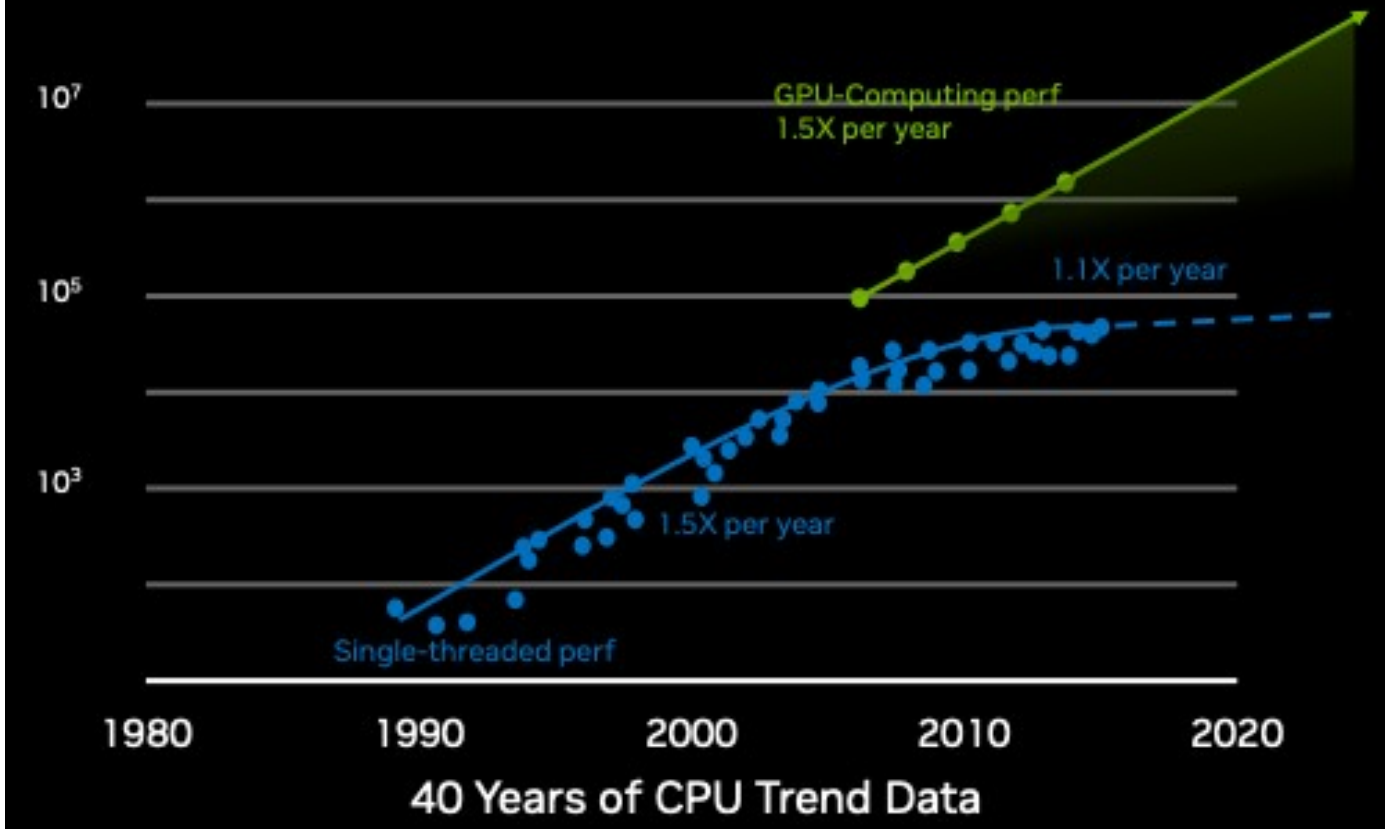


Tiling produced by SIVIA

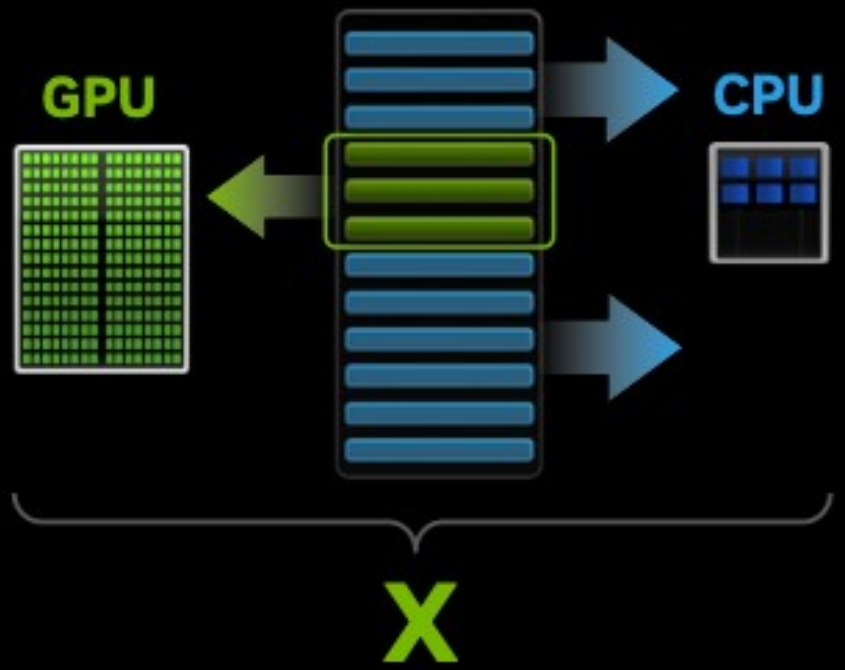
- But parallel resources are abundant
- Can interval methods be adapted to parallel computation?

French national GPU cluster
~ 86,344 CPUs
~ 2696 GPUs
~28 petaflops per s

Rise of GPU Computing

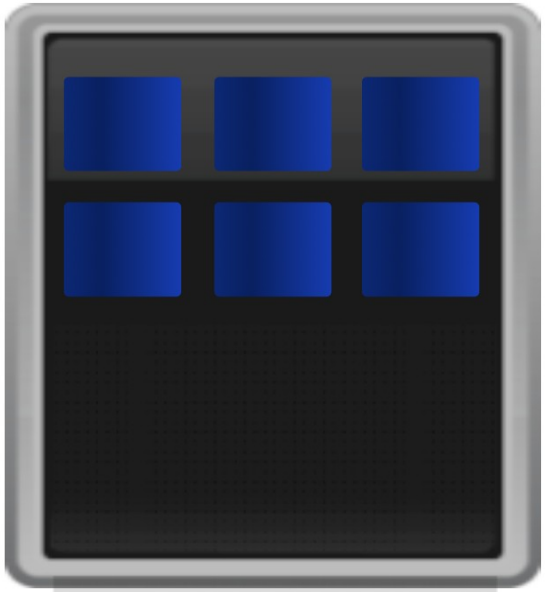


Original data up to the year 2010 collected and plotted by M. Horowitz,
F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten New plot and data collected for 2010-2015 by K. Rupp

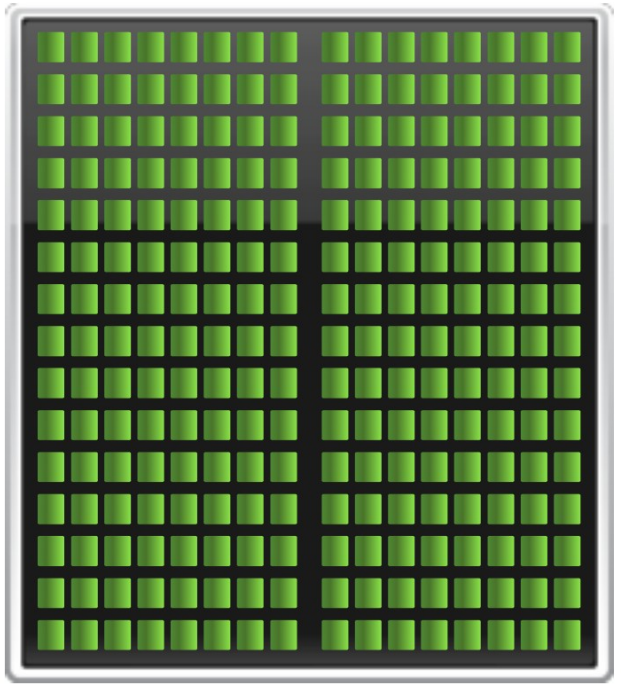


ACCELERATED COMPUTING

CPU
Optimized for
Serial Tasks

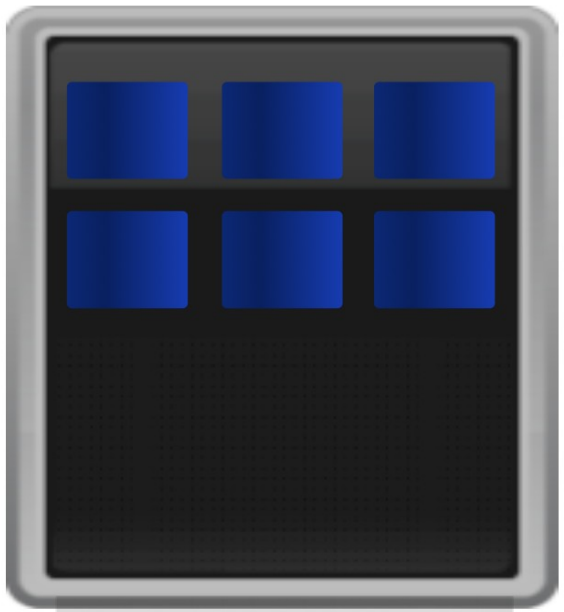


GPU Accelerator
Optimized for
Parallel Tasks



CPU

Optimized for Serial Tasks



CPU Strengths

- Very large main memory
- Very fast clock speeds
- Latency optimized via large caches
- Small number of threads can run very quickly

CPU Weaknesses

- Relatively low memory bandwidth
- Cache misses very costly
- Low performance/watt

GPU Strengths

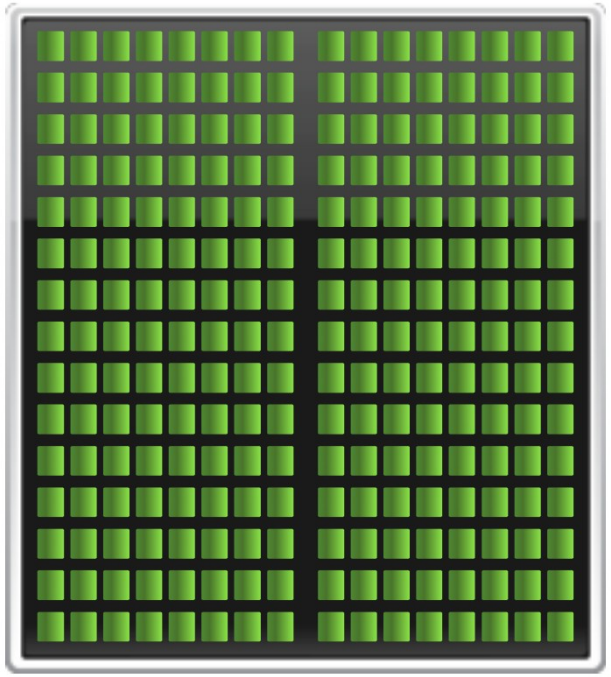
- High bandwidth main memory
- Significantly more compute resources
- Latency tolerant via parallelism
- High throughput
- High performance/watt

GPU Weaknesses

- Relatively low memory capacity
- Low per-thread performance

GPU Accelerator

Optimized for Parallel Tasks



SPEED V. THROUGHPUT

Speed

Throughput

Which is better depends on your needs…

GH100 GPU architecture

https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper

GH100 GPU a

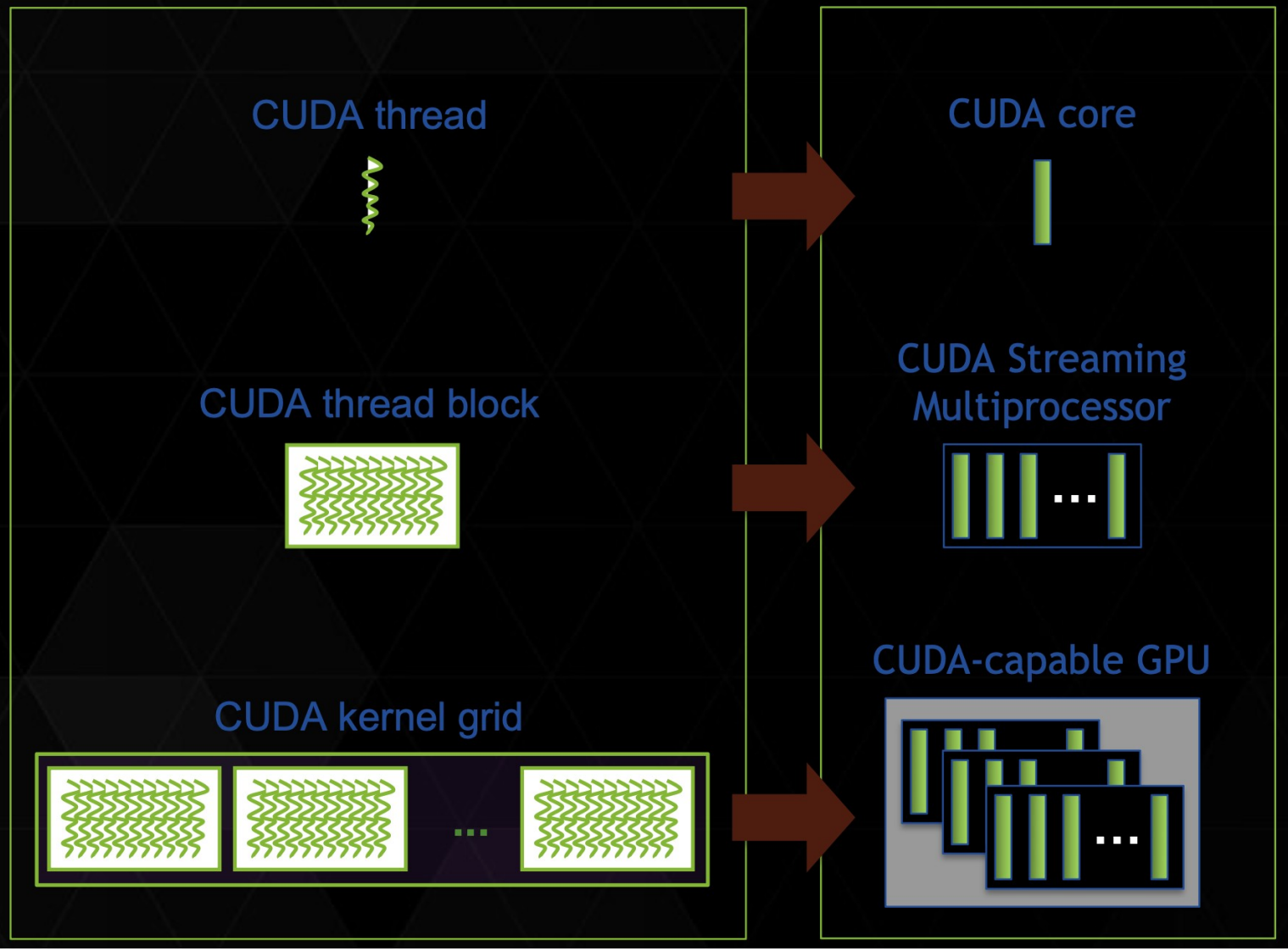
ht e/gtc22-whitepaper-hopper

GH100 GPU a

ht e/gtc22-whitepaper-hopper

- 132 of these in the H100 SXM5 GPU
- > 16000 FP32 cores in total
- We usually want number of threads >> num cores
- So we need **a lot** of threads!

KERNEL EXECUTION



- Each thread is executed by a core
- Each block is executed by one SM and does not migrate
- Several concurrent blocks can reside on one SM depending on the blocks' requirements and the SM's resources
- Each kernel is executed on one device
- Multiple kernels can execute on a device at one time

Julia + IntervalArithmetic.jl + CUDA.jl demo

Algorithm 2 Vector implementation of SIVIA algorithm (VSIVIA) algorithm

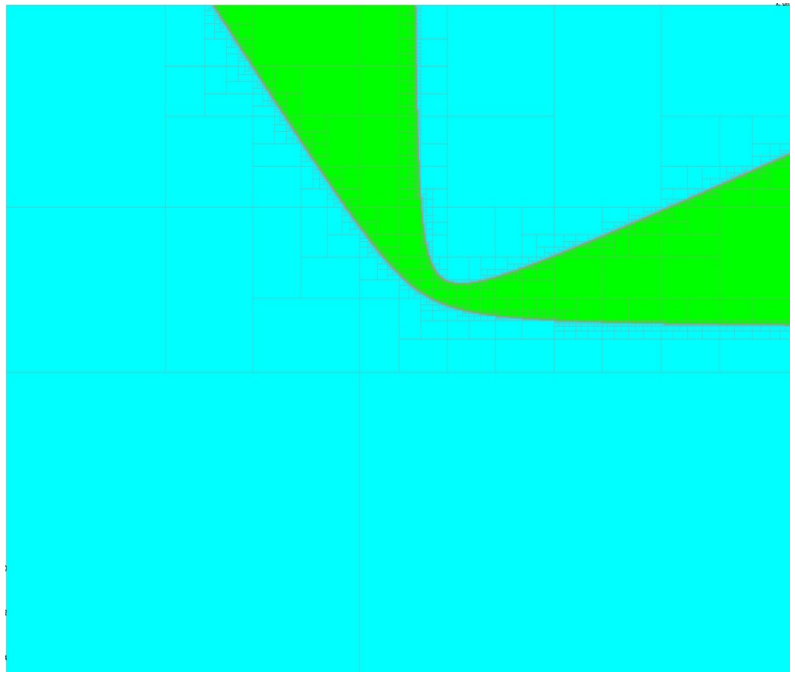
Require: • $\mathbb{Y} \subset \mathbb{R}^p$ • $[\mathbf{x_0}] \in \mathbb{IR}^n$ • $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^p$ • $\epsilon > 0$
Ensure: • $\mathcal{S}$, $\mathcal{N}$ and $\mathcal{E}$ such as: • $\mathcal{S} \subset (\mathbb{X} \cap [\mathbf{x_0}]) \subset \mathcal{S} \cup \mathcal{E}$ • $\mathcal{N} \cap \mathbb{X} = \emptyset$
• $width([\mathbf{x}]) < \epsilon \ (\forall [\mathbf{x}] \in \mathcal{E})$

1: **function** VSIVIA([**f**], $\mathbb{Y}$, [$\mathbf{x_0}$], ϵ)
2: $\mathcal{S} \leftarrow \mathcal{N} \leftarrow \mathcal{E} \leftarrow \emptyset$
3: $\mathcal{L} \leftarrow \{[\mathbf{x_0}]\}$
4: **while** $\mathcal{L} \neq \emptyset$ **do**
5: **in** $\leftarrow$ ([**f**]($\mathcal{L}$) $\subset \mathbb{Y}$) ▷ **in**: Vector of booleans same size as $\mathcal{L}$ (logical index)
6: **out** $\leftarrow$ ([**f**]($\mathcal{L}$) $\cap \mathbb{Y} = \emptyset$) ▷ **out**: Vector of booleans same size as $\mathcal{L}$ (logical index)
7: $push(\mathcal{S}, \mathcal{L}(\mathbf{in}))$ ▷ $push$: adds elements to a list
8: $push(\mathcal{N}, \mathcal{L}(\mathbf{out}))$
9: $\mathcal{U} \leftarrow \mathcal{L}(\neg\mathbf{in} \wedge \neg\mathbf{out})$ ▷ $\mathcal{U}$: Undecided boxes
10: **eps** $\leftarrow$ (**Width**($\mathcal{U}$) $< \epsilon$) ▷ **Width**: returns widths of largest intervals dimensions.
11: $push(\mathcal{E}, \mathcal{U}(\mathbf{eps}))$ ▷ **eps**: logical index
12: $\mathcal{L} \leftarrow$ **Bisect**($\mathcal{U}(\neg\mathbf{eps})$) ▷ **Bisect**: bisects each box in $\mathcal{U}$, returns list twice as long
13: **end while**
14: **return** $(\mathcal{S}, \mathcal{N}, \mathcal{E})$
15: **end function**

From “An Efficient Implementation of SIVIA Algorithm in a High-Level Numerical Programming Language ”, Herrero, Georgiou, Toumazou, Delaunay, Jaulin,

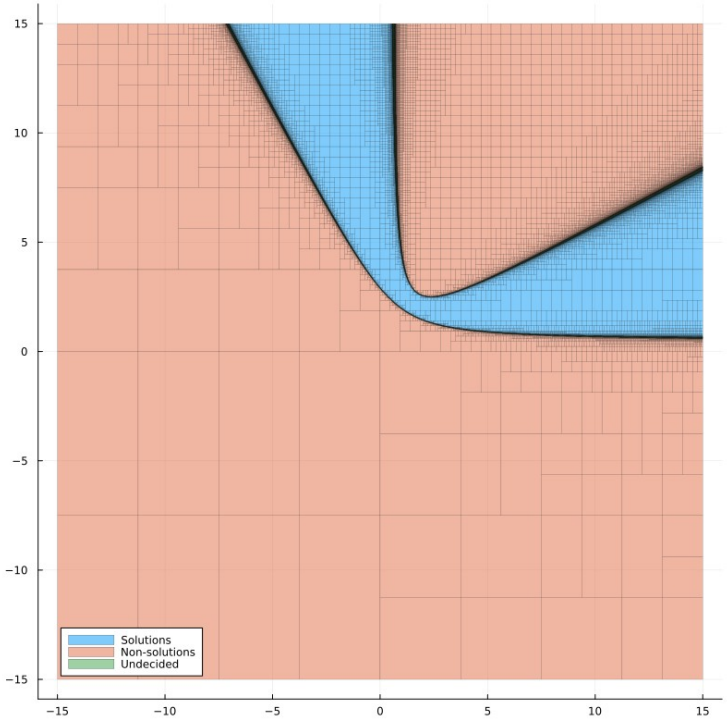
Localisation example

CODAC



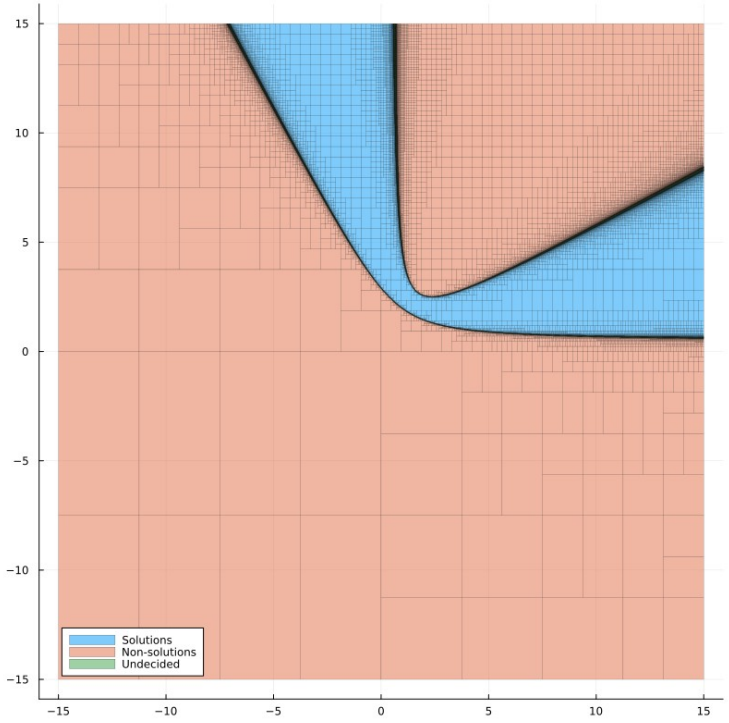
My laptop
~3.6s

IntervalArithmetic.jl



My laptop
~ 1.5s

IntervalArithmetic.jl + CUDA



NVIDIA V100
~0.014s

Error	CODAC	JULIA CPU	JULIA GPU	Speedup
0.01	3.19s	1.4s	0.016s	x87
0.001	27.97s	12.2s	0.044s	x277
0.0001	303.78s	238.1s	1.2s	X198.4